UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/676,743	09/30/2003	Anandhi Somasekaran	MSFT-2763/305222.1	7938

41505 7590 07/23/2008

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
07/23/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/676,743	SOMASEKARAN ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 April 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20,23-35 and 37-65* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20,23-35 and 37-65* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☒ Interview Summary (PTO-413) Paper No(s)/Mail Date. *4/15/2008*.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated April 28, 2008, responding to the Office action mailed January 28, 2008 provided in the rejection of claims 1-20, 23-35, and 37-65, wherein claims 1-4, 6, 12, 14-20, 26, 30-32, 35, 37, 40, 43-46, 50-52, 55-57, 60, and 63-65 were amended.

2. It is noticed that the current listing of pending claims is inadvertently listed as "1-20, 23-25 and 37-65". However, it should be corrected as --1-20, 23-35, and 37-65-- instead.

Claims 1-20, 23-35, and 37-65, remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Charisius et al.* - art made of record, as applied hereto.

3. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-7 and 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ulrich Roxburgh, (*BizTalk Orchestration™: Transactions, Exceptions, and Debugging, Feb. 2001, Microsoft™*) (hereinafter 'Roxburgh') in view of Charisius et al. (Pat. No. US 7,051,316 B2) (hereinafter 'Charisius' - art made of record)

5. **As to claim 1** (Currently Amended), Roxburgh discloses a business process service debugger for remotely debugging a business process service (e.g., P. 16, Sec. of "Debugging Components in Schedules"), comprising:

- means for reading stored state information regarding events related to at least one business process implemented for the business process service (e.g., Sec. of "Summary" - … implementing long-running business processes using

BizTalk Orchestration Services™; Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another; Sec. of "Tracing Schedules", 2nd Par. – when these COM+ applications execute a schedule, they generate various events that can be trapped and displayed; BizTalk Server provides a tool, called the XLANG™ Event Monitor, to trap and display these events; The XLANG™ Event Monitor can subscriber to events published by host applications on any number of distributed computers, and can store these events for later analysis);

Further, Roxburgh discloses the same way as debugging a standard COM+ component that is being called from a client application (e.g., Sec. of "Debugging Components in Schedules", 1st Par.), but does not explicitly disclose the following:

- means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service.
- means for displaying a symbolic representation of the operation of the business process service based on the stored state information; and
- means for remotely debugging the one business process service using the symbolic representation, communications connection and stored state information

However, in an analogous art of *Distributed Computing, Component System with Diagrammatic Graphical Representation of Code with Separate Delineated Display Area by Type*, Charisius discloses the following:

- means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 33, Lines 41-56 - ... may indicate the EJB target application server to the software development tool from a list of application servers that are retrieved from a configuration file and displayed by the software development tool ...; Col. 35, Lines 23-25 – The software development tool also receives an address port to listen for communications between the EJB target application server and the Client Application)
- means for displaying a symbolic representation of the operation of the business process service based on the stored state information (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....); and
- means for remotely debugging the one business process service using the symbolic representation, communications connection and stored state information (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target

application server; Col. 38, Lines 8-10 - … the platform hosting the EJB target application serer may be the remote computer)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Charisius into the Roxburgh's system to further provide the following in Roxburgh system:

- means for establishing a communications connection with a remote computer, wherein the remote computer is implementing the business process service.
- means for displaying a symbolic representation of the operation of the business process service based on the stored state information; and
- means for remotely debugging the one business process service using the symbolic representation, communications connection and stored state information

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Charisius' system which offers significant advantages to allow a developer to simultaneously view a graphical and a textual display of source code and to save significant programming development time as well as costs for conventional tools by allowing a developer to generate, compile, assemble, deploy, and debug a distributed computing component, such as an Enterprise JavaBean™, without having to use multiple conventional tools as once suggested by Charisius (e.g., Col. 4, Summary of the Invention, 1st and 2nd Pars.)

6. **As to claim 2** (Currently Amended) (incorporating the rejection in claim 1), Charisius discloses the business process service debugger, wherein business

processes and instances of the business process service other than those being debugged are not disrupted during debugging (e.g., Col. 50, Lines 13-17 - ... the software development tool allows the programmer to independently control the operation of the main test client and EJB in order to assess the impact of one on the operation of the other ...)

7. **As to claim 3** (Currently Amended) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the symbolic representation comprises a workflow of at least one business process in the business process service (e.g., the diagram on P. 8)

8. **As to claim 4** (Currently Amended) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger further comprising means for interacting with the business process service according to a user instruction (e.g., Sec. of "Debugging Schedules")

9. **As to claim 5** (Original) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the stored state information corresponds to a variable assignment within the business process service (e.g., Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another)

10. **As to claim 6** (Currently Amended) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the events are historical events that occurred prior to failure of the at least one business process (e.g., P. 14, Sec. of Tracing Schedules, 2nd Par. - … The XLANG™ Event Monitor can subscribe to events published by host applications on any number of distributed computers, and can store these events for later analysis)

11. **As to claim 7** (Previously Presented) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein the stored state information corresponds to message flow data and an order in which run time components performed the one business process as a result of message processing (e.g., Sec. of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another; P. 5, 1st Par. - … the implementation of the schedule reads a message from a transactional message queue and writes the message to another transactional message queue …)

12. **As to claim 9** (Previously Presented) (incorporating the rejection in claim 1), Charisius discloses the business process service debugger wherein the events are events that occur prior to an inserted breakpoint in the one business process (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein …; Fig. 24; Col. 26, Lines 59-64 - … includes state-management

callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....; Col. 47, Lines 56 through Col. 48, Line 10 - ... allows a programmer to set breakpoints in source code corresponding to the deployed EJB as well as set breakpoints in a client test program that resides with the software development tool ... By setting breakpoints ... at any line of source code corresponding to the deployed EJB or the client program ...)

13. **As to claim 10** (Original) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein said debugging means comprises means for detecting a location where the instance is being processed (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor)

14. **As to claim 11** (Original) (incorporating the rejection in claim 1), Roxburgh discloses the business process service debugger wherein said debugging means comprises means for detecting a location where the instance state is being stored (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor)

15. Claims 8, 12-20, and 23-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Roxburgh in view Charisius and further in view of Adams et al., (*BizTalk™ Unleashed 1st Edition, Feb. 2002, Sams Publishing*) (hereinafter 'Adams')

16. **As to claim 8** (Original) (incorporating the rejection in claim 1), Roxburgh discloses a transaction is an action or series of actions that transform a system from one consistent state to another (e.g., Sec, of "What Is a Transaction?"), but Roxburgh, and Charisius do not disclose the business process service debugger wherein said reading means further comprises means for reading stored business process service configuration information.

However, in an analogous art of *BizTalk™ Unleashed*, Adams discloses the business process service debugger wherein said reading means further comprises means for reading stored business process service configuration information (e.g., Sec. of "Developing Custom Tracking Solutions" - the two primary tracking activities: configuring tracking settings and viewing tracking data)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Roxburgh-Charisius' system to further provide the business process service debugger wherein said reading means further comprises means for reading stored business process service configuration information in Roxburgh-Charisius system.

The motivation is that it would further enhance the Roxburgh-Charisius' system by taking, advancing and/or incorporating Adams's system which offers significant

advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution")

17. **As to claim 12** (Currently amended), Roxburgh discloses a system for remotely debugging a distributed transactional application, comprising: a server (e.g., Sec. of "Introduction", 5th Par. – this article discusses methodologies to debug and troubleshoot Biztalk Server Orchestration™ Services and BizTalk™ Messaging Services), configured to execute an instance of a business process service, thereby generating runtime data (e.g., Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another)

Roxburgh discloses the same way as debugging a standard COM+ component that is being called from a client application (e.g., Sec. of "Debugging Components in Schedules", 1st Par.), but does not explicitly disclose a client computer configured to execute a debugging user interface (UI) process that establishes a communications connection with the server requests runtime data for at least one of the plurality of business processes, and generates a symbolic representation of the business service process showing any debugging break points specified by a user.

However, in an analogous art of *Distributed Computing, Component System with Diagrammatic Graphical Representation of Code with Separate Delineated Display Area by Type*, Charisius discloses a client computer configured to execute a debugging user interface (UI) process that establishes a communications connection with the server

requests runtime data for at least one of the plurality of business processes (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target application server; Col. 38, Lines 8-10 - ... the platform hosting the EJB target application serer may be the remote computer), and generates a symbolic representation of the business service process showing any debugging break points specified by a user (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Col. 47, Lines 56 through Col. 48, Line 10 - ... allows a programmer to set breakpoints in source code corresponding to the deployed EJB as well as set breakpoints in a client test program that resides with the software development tool ... By setting breakpoints ... at any line of source code corresponding to the deployed EJB or the client program ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Charisius into the Roxburgh's system to further provide a client computer configured to execute a debugging user interface (UI) process that establishes a communications connection with the server requests runtime data for at least one of the plurality of business processes, and generates a symbolic representation of the business service process showing any debugging break points specified by a user in the Roxburgh system.

The motivation is that it would further enhance the Roxburgh's system by taking, advancing and/or incorporating Charisius' system which offers significant advantages to allow a developer to simultaneously view a graphical and a textual display of source

code and to save significant programming development time as well as costs for conventional tools by allowing a developer to generate, compile, assemble, deploy, and debug a distributed computing component, such as an Enterprise JavaBean™, without having to use multiple conventional tools as once suggested by Charisius (e.g., Col. 4, Summary of the Invention, 1st and 2nd Pars.)

Furthermore, Roxburgh discloses a transaction is an action or series of actions that transform a system from one consistent state to another (e.g., Sec, of "What Is a Transaction?") and Charisius discloses allowing a developer to simultaneously view a graphical and a textual display of source code and to save significant programming development time as well as costs for conventional tools by allowing a developer to generate, compile, assemble, deploy, and debug a distributed computing component without having to use multiple conventional tools (e.g., Col. 4, Summary of the Invention, 1st and 2nd Pars.), but Roxburgh and Charisius do not explicitly disclose an interceptor for monitoring the runtime data and, when a specified break point is identified, causing the server to enter or leave a debugging state.

However, in an analogous art of *BizTalk™ Unleashed*, Adams discloses an interceptor for monitoring the runtime data and, when a specified break point is identified, causing the server to enter or leave a debugging state (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server

and returns a submission identifier; the submission ID should be displayed in the result window)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Roxburgh-Charisius' system to further provide an interceptor for monitoring the runtime data and, when a specified break point is identified, causing the server to enter or leave a debugging state in the Roxburgh-Charisius system.

The motivation is that it would further enhance the Roxburgh-Charisius' system by taking, advancing and/or incorporating Adams's system which offers significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution")

18. **As to claim 13** (Original) (incorporating the rejection in claim 12), Roxburgh discloses the system further comprising for storing business process service state information (e.g., Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another)

Further, Adams discloses a database for receiving the runtime data (e.g., Sec. of "Developing Custom tracking Solutions" – we begin by covering the tracking database, including the core tables where interchange data, document data, and routing data is stored)

19. **As to claim 14** (Currently Amended) (incorporating the rejection in claim 13), Roxburgh discloses the system further comprising a display device, for displaying the symbolic representation (e.g., P. 4, 2nd Par. – the transaction model for a schedule can be set by opening the properties dialog box for the Begin shape at the star of the schedule; Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another)

Further, Charisius discloses a user input device, wherein the input device is used to specify debugging break points (e.g., Col. 47, Lines 56 through Col. 48, Line 10 - ... allows a programmer to set breakpoints in source code corresponding to the deployed EJB as well as set breakpoints in a client test program that resides with the software development tool ... By setting breakpoints ... at any line of source code corresponding to the deployed EJB or the client program ...)

20. **As to claim 15** (Currently Amended) (incorporating the rejection in claim 14), Roxburgh discloses the system wherein the symbolic representation comprises presents a workflow representative of the program flow of the business process service (e.g., the diagram on P. 8)

21. **As to claim 16** (Currently Amended) (incorporating the rejection in claim 14), Roxburgh discloses the system wherein the display device further displays data representative of a message flow of the business process service (e.g., P. 5, the diagram; 1st Par. – the implementation of the schedule reads a message from

transactional message queue (i.e., receive queue) and write the message to another transaction message queue (i.e., send queue))

22. **As to claim 17** (Currently Amended) (incorporating the rejection in claim 14), Charisius discloses the system wherein the symbolic representation is presented according to stored state information (e.g., Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....)

23. **As to claim 18** (Currently Amended) (incorporating the rejection in claim 12), Adams discloses the system wherein a message box database is coupled between the server and client computer and is configured for communicating debugging requests from the client computer (e.g., Sec. of "BizTalk™ Messaging" – BizTalk™ Messaging is a set of services that provide a mechanism for routing documents in an enterprise environment; Sec. of "Configuration Objects Versus BizTalk™ Messaging Objects" – Biztalk™ Messaging objects are the actual objects stored in the Biztalk™ Messaging database)

24. **As to claim 19** (Currently Amended) (incorporating the rejection in claim 18), Adams discloses the system wherein the UI process comprises an application program interface for communicating with the message box database (e.g., Sec. of "BizTalk™

Messaging" – BizTalk™ Messaging is a set of services that provide a mechanism for routing documents in an enterprise environment; Sec. of "Configuration Objects Versus BizTalk™ Messaging Objects" – Biztalk™ Messaging objects are the actual objects stored in the Biztalk™ Messaging database)

25. **As to claim 20** (Currently Amended) (incorporating the rejection in claim 18), Adams discloses the system further comprising a tracking database to receive business process service tracking information (e.g., Sec. of "Developing Custom tracking Solutions" – we begin by covering the tracking database, including the core tables where interchange data, document data, and routing data is stored), and further Charisius discloses wherein the UI process comprises a UI component for communicating with the tracking database (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Fig. 20, element 2008 - DBMS)

26. **As to claim 23** (Original) (incorporating the rejection in claim 12), Adams discloses the system wherein the interceptor is a component of a computer language that provides stored state tracking information (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking

Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window)

27. **As to claim 24** (Original) (incorporating the rejection in claim 12), Charisius discloses the system wherein the UI process detects a location where the instance is being processed (e.g., Col. 47, Lines 56 through Col. 48, Line 10 - ... allows a programmer to set breakpoints in source code corresponding to the deployed EJB as well as set breakpoints in a client test program that resides with the software development tool ... By setting breakpoints ... at any line of source code corresponding to the deployed EJB or the client program ...)

28. **As to claim 25** (Original) (incorporating the rejection in claim 12), Charisius discloses the system wherein the UI process detects a location where the instance state is being stored (e.g., Col. 47, Lines 56 through Col. 48, Line 10 - ... allows a programmer to set breakpoints in source code corresponding to the deployed EJB as well as set breakpoints in a client test program that resides with the software development tool ... By setting breakpoints ... at any line of source code corresponding to the deployed EJB or the client program ...)

Claim Rejections – 35 USC § 102(e)

The following is quotation of 35 U.S.C. 102(e) which form the basis for all obviousness rejections set forth in this office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for

patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

29. Claims 40-41, 43-45, 60-61, and 63-65 are rejected under 35 U.S.C. 102(e) as being anticipated by Charisius

30. **As to claim 40** (Currently Amended), Charisius discloses a method in a computer system for displaying on a display device a graphical debugging environment for a business process service (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 32, Lines 38-39 – Compiling, Deploying and Debugging an Enterprise JavaBean™; Lines 53-59 - ... the software development tools allows the programmer to deploy and test the EJB. Figs. 38A-38F depict a flowchart illustrating an exemplary process performed by the software development tool to compile, deploy, and debug an EJB) , the method comprising:

- obtaining design information about the business process service (e.g., Col. 36, Lines 4-9 - ... the software development tool parses a configuration associated with the EJB target application server for an identification of the EJB specification that the EJB target application server supports ...);
- obtaining tracking information about execution of the business process service (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target application server; Col. 35, Lines 23-25 – The software development tool also receives an address

port to listen for communications between the EJB target application server and the Client Application);

- generating symbolic representation of the operation of the business process service according to the design information and tracking information (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....); and
- displaying on the display device a graphical debugging environment showing the symbolic representation (e.g., Col. 28, Line 46 through Col. 29, Line 3 - The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...)

31. **As to claim 41** (Original) (incorporating the rejection in claim 40), Charisius discloses further comprising receiving runtime data for the business process service and presenting the runtime data on the display device (e.g., Col. 28, Line 46 through Col. 29, Line 3 - The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...)

32. **As to claim 43** (Currently Amended) (incorporating the rejection in claim 40), Charisius discloses wherein the graphical debugging environment enables a user to place a breakpoint in the symbolic representation of the operation of the business process service (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein …; Fig. 24; Col. 26, Lines 59-64 - … includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....)

33. **As to claim 44** (Currently Amended) (incorporating the rejection in claim 40), Charisius discloses the symbolic representation comprising symbols, wherein the graphical debugging environment also displays information about the symbols (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein …; Fig. 24; Col. 26, Lines 59-64 - … includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....)

34. **As to claim 45** (Currently Amended) (incorporating the rejection in claim 40), Charisius discloses further comprising receiving input from an input device to place a

break point proximate a symbol, and presenting a symbol representing the break point on the symbolic representation (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...)

35. **As to claim 60** (Currently Amended), Charisius discloses a computer-readable storage medium having computer-executable instructions for performing a method for displaying on a display device a graphical debugging environment for a business process service (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 32, Lines 38-39 – Compiling, Deploying and Debugging an Enterprise JavaBean™; Lines 53-59 - ... the software development tools allows the programmer to deploy and test the EJB. Figs. 38A-38F depict a flowchart illustrating an exemplary process performed by the software development tool to compile, deploy, and debug an EJB), the method comprising:

- obtaining design information about the business process service (e.g., Col. 36, Lines 4-9 - ... the software development tool parses a configuration associated with the EJB target application server for an identification of the EJB specification that the EJB target application server supports ...);
- obtaining configuration information about the business process service (e.g., Col. 36, Lines 4-9 - ... the software development tool parses a configuration

associated with the EJB target application server for an identification of the EJB specification that the EJB target application server supports ...);

- generating symbolic representation of the operation of the business process service according to the design information and configuration information (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....); and
- displaying on the display device a graphical debugging environment showing the symbolic representation (e.g., Col. 28, Line 46 through Col. 29, Line 3 - The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...)

36. **As to claim 61** (Currently Amended) (incorporating the rejection in claim 60), please refer to claim **41** as set forth accordingly.

37. **As to claims 63-65**, refer to above **claims 43-45**, accordingly.

38. Claims 26, 42, 46-59, and 62 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius in view Adams

39. **As to claim 26** (Currently Amended), Charisius discloses a method for debugging an instance of a business process service running on a remote computer (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 32, Lines 38-39 – Compiling, Deploying and Debugging an Enterprise JavaBean™; Lines 53-59 - … the software development tools allows the programmer to deploy and test the EJB. Figs. 38A-38F depict a flowchart illustrating an exemplary process performed by the software development tool to compile, deploy, and debug an EJB), comprising:

- generating for display, in a graphical user interface (GUI), a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process service (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein …; Fig. 24; Col. 26, Lines 59-64 - … includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....);
- receiving a debugging command generated by a user interacting with the GUI (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target application server);

- a direct client connection channel with the remote compute (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 33, Lines 41-56 - ... may indicate the EJB target application server to the software development tool from a list of application servers that are retrieved from a configuration file and displayed by the software development tool ...; Col. 35, Lines 23-25 – The software development tool also receives an address port to listen for communications between the EJB target application server and the Client Application);
- receiving a runtime request, generated by a user interacting with the GUI, for event information generated by execution of the instance of the business process service (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....);
- sending the runtime request to the remote computer for processing by the remote computer (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target application server)

Further, Charisius discloses allowing a developer to simultaneously view a graphical and a textual display of source code and to save significant programming development time as well as costs for conventional tools by allowing a developer to generate, compile, assemble, deploy, and debug a distributed computing component without having to use multiple conventional tools (e.g., Col. 4, Summary of the Invention, 1st and 2nd Pars.), but does not explicitly disclose causing an interceptor to monitor runtime data generated by the instance of the business process in accordance with the debugging command.

However, in an analogous art of *BizTalk™ Unleashed*, Adams discloses causing an interceptor to monitor runtime data generated by the instance of the business process in accordance with the debugging command (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Charisius' system to further provide causing an interceptor to monitor runtime data generated by the instance of the business process in accordance with the debugging command in the Charisius system.

The motivation is that it would further enhance the Charisius' system by taking, advancing and/or incorporating Adams's system which offers significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution")

40. **As to claim 42** (Previously Presented) (incorporating the rejection in claim 41), Adams discloses wherein the runtime data comprises historical message flow information including identification of run time messages that were constructed as a result of processing received messages, and further comprises order information pertaining to the order in which different run time components were executed as a result of processing received messages (e.g., Chapter 10 – Using the BizTalk Orchestration Designer, 2nd Par. – the BizTalk Orchestration Designer contains wizards that guide you through creating and configuring ports and message flows; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window)

41. **As to claim 46** (Currently Amended), Charisius discloses a computer-readable storage medium having computer-executable instructions for performing a method for debugging an instance of a business process service running on a remote computer (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 32, Lines 38-39 –

Compiling, Deploying and Debugging an Enterprise JavaBean™; Lines 53-59 - ... the software development tools allows the programmer to deploy and test the EJB; Figs. 38A-38F depict a flowchart illustrating an exemplary process performed by the software development tool to compile, deploy, and debug an EJB), comprising:

- generating for display, in a graphical user interface (GUI), a symbolic representation of the business process service based on a correlation of information about the design and execution of the business process services (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....);
- receiving a debugging command generated by a user interacting with the GUI (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target application server);
- establishing a direct client connection channel with the remote computer (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 33, Lines 41-56 - ... may indicate the EJB target application server to the software

development tool from a list of application servers that are retrieved from a configuration file and displayed by the software development tool ...);

- receiving a runtime request; and sending the runtime request to the remote computer for processing by the remote computer (Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....)

Further, Charisius discloses allowing a developer to simultaneously view a graphical and a textual display of source code and to save significant programming development time as well as costs for conventional tools by allowing a developer to generate, compile, assemble, deploy, and debug a distributed computing component without having to use multiple conventional tools (e.g., Col. 4, Summary of the Invention, 1st and 2nd Pars.), but does not explicitly disclose causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command.

However, in an analogous art of *BizTalk™ Unleashed*, Adams discloses causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring

tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Adams into the Charisius' system to further provide causing an interceptor to monitor runtime data generated by the instance of the business process service in accordance with the debugging command in Charisius system.

The motivation is that it would further enhance the Charisius' system by taking, advancing and/or incorporating Adams's system which offers significant advantages for building customized tracking solutions using the infrastructure provided by Biztalk server as once suggested by Adams (e.g., Sec. of "Developing Custom Tracking Solution")

42. **As to claims 47-55**, refer to above **claims 27-35**, accordingly.

43. **As to claim 56** (Currently Amended) (incorporating the rejection in claim 30), Charisius discloses the computer-readable storage medium, wherein the debugging command is a request for data regarding an instance of the business process service (e.g., Fig. 61 - an exemplary user interface displayed by the software development tool in Fig. 2 for debugging the EJB on the target application server)

44. **As to claims 57-59**, refer to above **claims 37-39**, accordingly.

45. **As to claim 62**, refer to above **claims 42**, accordingly.

46. Claims 27-35, and 37-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius in view of Adams and further in view of Roxburgh

47. **As to claim 27** (Original) (incorporating the rejection in claim 26), Adams discloses further comprising: querying a database containing a status of the business process service; displaying a query result on a display device; receiving user input with respect to the query result (e.g., Sec. of "Tracking Events in the Tracking Database", 1st Par. – the BizTalk™ Document Tracking Web application all you to query the database for document flows between organizations or applications; the XLANG™ events captured in the database can be viewed in relationship with the message flows displayed by document tracking Web application)

Further, Charisius discloses establishing the direct client connection channel in response to the user input (e.g., Fig. 40 – an exemplary user interface displayed by the software development tool for receiving an indication of an EJB target application server; Col. 33, Lines 41-56 - ... may indicate the EJB target application server to the software development tool from a list of application servers that are retrieved from a configuration file and displayed by the software development tool ...)

48. **As to claim 28** (Original) (incorporating the rejection in claim 27), Roxburgh discloses the information contained in the database is instance runtime data (e.g., Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another)

49. **As to claim 29** (Original) (incorporating the rejection in claim 27), Adams discloses the information contained in the database is instance tracking data (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window)

50. **As to claim 30** (Currently Amended) (incorporating the rejection in claim 26), Roxburgh discloses further comprising: creating the business process service using a process designer (e.g., Sec. of "Introduction", 1st Par. – with Microsoft BizTalk Orchestration Designer™, uses can design long-running business processes, specify an implementation for the individual actions that make up those processes, and compile this information into an executable XML representation, known as an XLANG™ schedule; the schedules created are distributed across time, organizations, and applications, in a loosely coupled and scalable manner)

Further, Adams discloses saving a business process service configuration and symbolic data in a database as information about the design of the business process service (e.g., Sec. of "Developing Custom Tracking Solution" – to perform the two primary tracking activities: configuring tracking settings and viewing tracking data; building customized tracking solutions using the infrastructure provided by BizTalk™ Server; Sec. of "Example: A Custom Tracking Solution", 4th Par. – this submits an interchange to BizTalk™ Server and returns a submission identifier; the submission ID should be displayed in the result window); Furthermore, Adams discloses displaying the symbolic representation on a display device according to the saved business process service configuration and symbolic data; generating a runtime request based on the symbolic representation; and displaying a result of the runtime request on the display device (e.g., Col. 28, Line 46 through Col. 29, Line 3 – The software development tool displays a symbol with a display area to identify the corresponding method type or reference type displayed therein ...; Fig. 24; Col. 26, Lines 59-64 - ... includes state-management callback methods (depicted graphically as 2404) that are invoked by the container 2150 to notify the EJB EntityBean™ when certain events are to occur on the EJB Application Server 2148 ....)

51. **As to claim 31** (Currently Amended) (incorporating the rejection in claim 30), Roxburgh discloses the symbolic representation comprises a shape corresponding to an operation in the business process service (e.g., the diagram on P. 8)

52. **As to claim 32** (Currently Amended) (incorporating the rejection in claim 30), Roxburgh discloses the symbolic representation comprises a workflow representation of the business process service (e.g., the diagram on P. 8)

53. **As to claim 33** (Original) (incorporating the rejection in claim 30, Roxburgh discloses the saving step takes place in connection with compiling and deploying the business process service (e.g., Sec. of "Introduction", 1st Par. – with Microsoft BizTalk Orchestration Designer™, uses can design long-running business processes, specify an implementation for the individual actions that make up those processes, and compile this information into an executable XML representation, known as an XLANG™ schedule; the schedules created are distributed across time, organizations, and applications, in a loosely coupled and scalable manner)

54. **As to claim 34** (Original) (incorporating the rejection in claim 30), Roxburgh discloses the business process service is implemented in a computer language that provides stored state information (e.g., Sec. of "Introduction", 1st Par. – with Microsoft BizTalk Orchestration Designer™, uses can design long-running business processes, specify an implementation for the individual actions that make up those processes, and compile this information into an executable XML representation, known as an XLANG™ schedule; the schedules created are distributed across time, organizations, and applications, in a loosely coupled and scalable manner)

55. **As to claim 35** (Currently Amended) (incorporating the rejection in claim 26), Charisius discloses the debugging command is a break point (e.g., Col. 47, Lines 56 through Col. 48, Line 10 - ... allows a programmer to set breakpoints in source code corresponding to the deployed EJB as well as set breakpoints in a client test program that resides with the software development tool ... By setting breakpoints ... at any line of source code corresponding to the deployed EJB or the client program ...; Col. 48, Lines 5-10 – The group of execution control commands includes ...)

56. **As to claim 37** (Currently Amended) (incorporating the rejection in claim 26), Roxburgh discloses the runtime data is state information (e.g., Sec, of "What Is a Transaction?", 3rd Par. – a transaction is an action or series of actions that transform a system from one consistent state to another)

57. **As to claim 38** (Original) (incorporating the rejection in claim 26), Roxburgh discloses further comprising detecting a location where the instance is being processed (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor)

58. **As to claim 39** (Original) (incorporating the rejection in claim 26), Roxburgh discloses further comprising detecting a location where an instance state is being stored (e.g., P. 15, last Par. – each instance in addition has the unique identifier for the

instance listed (the instance GUID); any of the listed running schedule instances can be suspended or terminated from the XLANG™ Event Monitor)

Conclusion

59. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192
July 17, 2008

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192